

---

# **f5-icontrol-rest Documentation**

***Release 1.0.5***

**F5 Networks**

April 01, 2016



<b>1</b>	<b>Overview</b>	<b>1</b>
<b>2</b>	<b>Installation</b>	<b>3</b>
2.1	Using Pip . . . . .	3
2.2	GitHub . . . . .	3
<b>3</b>	<b>Examples</b>	<b>5</b>
<b>4</b>	<b>Module Documentation</b>	<b>7</b>
4.1	icontrol . . . . .	7
4.1.1	icontrol package . . . . .	7
	Submodules . . . . .	7
	icontrol.session module . . . . .	7
	Module contents . . . . .	10
<b>5</b>	<b>Contact</b>	<b>11</b>
<b>6</b>	<b>License</b>	<b>13</b>
6.1	Apache V2.0 . . . . .	13
6.2	Contributor License Agreement . . . . .	13
	<b>Python Module Index</b>	<b>15</b>



---

### Overview

---

The F5 Networks® *icontrol* module is used to send commands to the BIGIP® iControl® REST API. The library maintains a HTTP session (which is a `requests.Session`) and does URL validation and logging.



---

## Installation

---

### 2.1 Using Pip

```
$ pip install f5-icontrol-rest
```

### 2.2 GitHub

[F5Networks/f5-icontrol-rest-python](#)





---

**Examples**

---

```
from iconcontrol.session import iControlRESTSession
icr_session = iControlRESTSession('myuser', 'mypass')

# GET to https://bigip.example.com/mgmt/tm/ltn/nat/~Common~mynat
icr_session.get(
    'https://bigip.example.com/mgmt/tm/ltn/nat',
    name='mynat',
    partition='Common')

# GET to https://bigip.example.com/mgmt/tm/ltn/nat
icr_session.get('https://bigip.example.com/mgmt/tm/ltn/nat')

# POST with json data
icr_session.post('https://bigip.example.com/mgmt/tm/ltn/nat', \
    json={'name': 'myname', 'partition': 'Common'})
```



---

## Module Documentation

---

### 4.1 icontrol

#### 4.1.1 icontrol package

##### Submodules

##### icontrol.session module

A BigIP-RESTServer URI handler. REST-APIs use it on the `requests` library.

Use this module to make calls to a BigIP-REST server. It will handle:

1. URI Sanitization uri's produced by this module are checked to ensure compliance with the BigIP-REST server interface
2. Session Construction – the `iControlRESTSession` wraps a `requests.Session` object.
3. Logging – pre- and post- request state is logged.
4. Exception generation – Errors in URL construction generate `BigIPInvalidURL` subclasses; unexpected HTTP status codes raise `iControlUnexpectedHTTPError`.

The core functionality of the module is implemented via the `iControlRESTSession` class. Calls to its' HTTP-methods are checked, pre-logged, submitted, and post-logged.

There are 2 modes of operation “full\_uri”, and “uri\_as\_parts”, toggled by the `uri_as_parts` boolean keyword param that can be passed to methods. It defaults to `False`. Use `uri_as_parts` when you want to leverage the full functionality of this library, and have it construct your uri for you. Example Use in `uri_as_parts` mode:

```
>>> iCRS = iControlRESTSession('jrandomhacker', 'insecure')
>>> iCRS.get('https://192.168.1.1/mgmt/tm/ltnat/', partition='Common', name='VALIDNAME', uri_as_parts=True)
```

In `full_uri` mode:

```
>>> iCRS.get('https://192.168.1.1/mgmt/tm/ltnat/~Common~VALIDNAME')
```

NOTE: If used via the `f5-common-python` library the typical mode is “full\_uri” since that library binds uris to Python objects.

Available functions:

- `iCRS.{get, post, put, delete, patch}`: `requests.Session.VERB` wrappers
- `decorate_HTTP_verb_method`: this function preps, logs, and handles requests

against the BigIP REST Server, by pre- and post- processing the above methods.

**exception** `icontrol.session.BigIPInvalidURL`

Bases: `exceptions.Exception`

**exception** `icontrol.session.InvalidBigIP_ICRURI`

Bases: `icontrol.session.BigIPInvalidURL`

**exception** `icontrol.session.InvalidInstanceNameOrFolder`

Bases: `icontrol.session.BigIPInvalidURL`

**exception** `icontrol.session.InvalidPrefixCollection`

Bases: `icontrol.session.BigIPInvalidURL`

**exception** `icontrol.session.InvalidScheme`

Bases: `icontrol.session.BigIPInvalidURL`

**exception** `icontrol.session.InvalidSuffixCollection`

Bases: `icontrol.session.BigIPInvalidURL`

`icontrol.session.decorate_HTTP_verb_method` (*method*)

Prepare and Post-Process HTTP VERB method for BigIP-RESTServer request.

This function decorates all of the HTTP VERB methods in the `iControlRESTSession` class. It provides the core logic for this module. If necessary it validates and assembles a uri from parts with a call to `generate_bigip_uri`.

Then it:

- 1.pre-logs the details of the request
- 2.submits the request
- 3.logs the response, included expected status codes
- 4.raises exceptions for unexpected status codes. (i.e. not doc'd as BigIP RESTServer codes.)

`icontrol.session.generate_bigip_uri` (*base\_uri, partition, name, suffix, \*\*kwargs*)

(str, str, str) → str

This function checks the supplied elements to see if each conforms to the specification for the appropriate part of the URI. These validations are conducted by the helper function `_validate_uri_parts`. After validation the parts are assembled into a valid BigIP REST URI string which is then submitted with appropriate metadata.

```
>>> generate_bigip_uri('https://0.0.0.0/mgmt/tm/ltnat/', 'CUSTOMER1', 'nat52', params={'a':1})
'https://0.0.0.0/mgmt/tm/ltnat/~CUSTOMER1~nat52'
>>> generate_bigip_uri('https://0.0.0.0/mgmt/tm/ltnat/', 'CUSTOMER1', 'nat52', params={'a':1}, suffix='/wacky')
'https://0.0.0.0/mgmt/tm/ltnat/~CUSTOMER1~nat52/wacky'
>>> generate_bigip_uri('https://0.0.0.0/mgmt/tm/ltnat/', '', '', params={'a':1}, suffix='/thwacky')
'https://0.0.0.0/mgmt/tm/ltnat/thwacky'
```

**class** `icontrol.session.iControlRESTSession` (*username, password, \*\*kwargs*)

Bases: `object`

Represents a `requests.Session` that communicates with a BigIP.

Instantiate one of these when you want to communicate with a BigIP-REST Server, it will handle BigIP-specific details of the uri's. In the `f5-common-python` library, an `iControlRESTSession` is instantiated during BigIP instantiation and associated with it as an attribute of the BigIP (a compositional vs. inheritable association).

Objects instantiated from this class provide an HTTP 1.1 style session, via the `requests.Session` object, and HTTP-methods that are specialized to the BigIP-RESTServer interface.

**delete** (*RIC\_base\_uri*, *\*\*kwargs*)

Sends a HTTP DELETE command to the BIGIP REST Server.

Use this method to send a DELETE command to the BIGIP. When calling this method with the optional arguments *name* and *partition* as part of *\*\*kwargs* they will be added to the *uri* passed in separated by ~ to create a proper BIGIP REST API URL for objects.

All other parameters passed in as *\*\*kwargs* are passed directly to the `requests.Session.delete()`

#### Parameters

- **uri** (*str*) – A HTTP URI
- **name** (*str*) – The object name that will be appended to the uri
- **partition** (*str*) – The partition name that will be appened to the uri
- **\*\*kwargs** – The `requests.Session.delete()` optional params

**get** (*RIC\_base\_uri*, *\*\*kwargs*)

Sends a HTTP GET command to the BIGIP REST Server.

Use this method to send a GET command to the BIGIP. When calling this method with the optional arguments *name* and *partition* as part of *\*\*kwargs* they will be added to the *uri* passed in separated by ~ to create a proper BIGIP REST API URL for objects.

All other parameters passed in as *\*\*kwargs* are passed directly to the `requests.Session.get()`

#### Parameters

- **uri** (*str*) – A HTTP URI
- **name** (*str*) – The object name that will be appended to the uri
- **partition** (*str*) – The partition name that will be appened to the uri
- **\*\*kwargs** – The `requests.Session.get()` optional params

**patch** (*RIC\_base\_uri*, *\*\*kwargs*)

Sends a HTTP PATCH command to the BIGIP REST Server.

Use this method to send a PATCH command to the BIGIP. When calling this method with the optional arguments *name* and *partition* as part of *\*\*kwargs* they will be added to the *uri* passed in separated by ~ to create a proper BIGIP REST API URL for objects.

All other parameters passed in as *\*\*kwargs* are passed directly to the `requests.Session.patch()`

#### Parameters

- **uri** (*str*) – A HTTP URI
- **data** (*str*) – The data to be sent with the PATCH command
- **name** (*str*) – The object name that will be appended to the uri
- **partition** (*str*) – The partition name that will be appened to the uri
- **\*\*kwargs** – The `requests.Session.patch()` optional params

**post** (*RIC\_base\_uri*, *\*\*kwargs*)

Sends a HTTP POST command to the BIGIP REST Server.

Use this method to send a POST command to the BIGIP. When calling this method with the optional arguments *name* and *partition* as part of *\*\*kwargs* they will be added to the *uri* passed in separated by ~ to create a proper BIGIP REST API URL for objects.

All other parameters passed in as `**kwargs` are passed directly to the `requests.Session.post()`

#### Parameters

- **uri** (*str*) – A HTTP URI
- **data** (*str*) – The data to be sent with the POST command
- **json** (*dict*) – The JSON data to be sent with the POST command
- **name** (*str*) – The object name that will be appended to the uri
- **partition** (*str*) – The partition name that will be appended to the uri
- **\*\*kwargs** – The `requests.Session.post()` optional params

**put** (*RIC\_base\_uri*, *\*\*kwargs*)

Sends a HTTP PUT command to the BIGIP REST Server.

Use this method to send a PUT command to the BIGIP. When calling this method with the optional arguments `name` and `partition` as part of `**kwargs` they will be added to the `uri` passed in separated by `~` to create a proper BIGIP REST API URL for objects.

All other parameters passed in as `**kwargs` are passed directly to the `requests.Session.put()`

#### Parameters

- **uri** (*str*) – A HTTP URI
- **data** (*str*) – The data to be sent with the PUT command
- **json** (*dict*) – The JSON data to be sent with the PUT command
- **name** (*str*) – The object name that will be appended to the uri
- **partition** (*str*) – The partition name that will be appended to the uri
- **\*\*kwargs** – The `requests.Session.put()` optional params

**exception** `icontrol.session.iControlUnexpectedHTTPError` (*\*args*, *\*\*kwargs*)

Bases: `requests.exceptions.HTTPError`

## Module contents

---

## Contact

---

Join us on [Slack!](#)





---

**License**

---

## **6.1 Apache V2.0**

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## **6.2 Contributor License Agreement**

Individuals or business entities who contribute to this project must have completed and submitted the [F5® Contributor License Agreement](#) to [Openstack\\_CLA@f5.com](mailto:Openstack_CLA@f5.com) prior to their code submission being included in this project.



**i**

`icontrol`, [10](#)  
`icontrol.session`, [7](#)



## B

BigIPInvalidURL, 8

## D

decorate\_HTTP\_verb\_method() (in module icon-  
trol.session), 8

delete() (icontrol.session.iControlRESTSession method),  
8

## G

generate\_bigip\_uri() (in module icontrol.session), 8

get() (icontrol.session.iControlRESTSession method), 9

## I

icontrol (module), 10

icontrol.session (module), 7

iControlRESTSession (class in icontrol.session), 8

iControlUnexpectedHTTPError, 10

InvalidBigIP\_ICRURI, 8

InvalidInstanceNameOrFolder, 8

InvalidPrefixCollection, 8

InvalidScheme, 8

InvalidSuffixCollection, 8

## P

patch() (icontrol.session.iControlRESTSession method),  
9

post() (icontrol.session.iControlRESTSession method), 9

put() (icontrol.session.iControlRESTSession method), 10